8-21-2018

Using Packet Timing Information in Website Fingerprinting

Mohammad Saidur Rahman

Using Packet Timing Information in Website Fingerprinting

by

Mohammad Saidur Rahman

Committe Members

Dr. Matthew Wright

Bill Stackpole

Dr. Leon Reznik

In partial fulfillment of the requirements for the degree of

Master of Science in Computing Security

Rochester Institute of Technology

B. Thomas Golisano College of Computing & Information Sciences

Department of Computing Security

August 21, 2018

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

Master of Science in
Computing Security

Thesis Approval Form

Student Name: Mohammad Saidur Rahman

Thesis Title: Using Packet Timing Information in Website Fingerprinting

Thesis Committee

Name	Signature	Date
Dr. Matthew Wright		
Committee Chair		
Bill Stackpole		
Committee Member		
Dr. Leon Reznik		
Committee Member		

To my Parents and All the Teachers Who Made me Who I am Today!

Acknowledgments

I would like to express my gratitude to all the teachers in my life who have taught, inspired, and supported me till now. I am and always will be grateful to them. I owe all of my professional accomplishments to them.

Firstly, I owe my life time service to make my parents and teachers proud. Only being thankful to my parents is not enough who taught me being a better human being with value sense.

Secondly, I am profoundly grateful to my adviser Professor Dr. Matthew Wright for his continuous inspiration, encouragement, feedback, and invaluable advice throughout the course of my thesis work. He is one of the great examples in my life as a great professor and as a great human being. I look forward to learning more and more in the next phases of my life.

I would also like to express my gratitude to my committee members Professor Bill Stackpole, and Professor Dr. Leon Reznik for their continuous feedback, suggestions, and comments on my thesis work.

Next, I would like to express my heartiest thankfulness to those who taught, inspired, encouraged, and supported me to pursue my higher education. The whole list would be far too long. I include the few who are directly related to this endeavor from the MIS department of University of Dhaka: Md. Ariful Islam, Dr. Hasibur Rashid, Dr. K. M. Salah Uddin, and Dr. Md. Rakibul Hoque.

The last but not the least, I want to express my thankfulness to my two great mentors, and friends from the Center for Cybersecurity: Payap Sirinam and Mohsen Imani for their continuous mentoring, support, care, and love. I would also like to thank Kantha Girish Gangadhara for supporting me in my project and being a great friend.

Abstract

Using Packet Timing Information in Website Fingerprinting

Mohammad Saidur Rahman

Rochester Institute of Technology

Supervisor: Dr. Matthew Wright

Website Fingerprinting (WF) enables an eavesdropper to discover what sites the user is visiting despite the use of a VPN or even the Tor anonymity system. Recent WF attacks on Tor have reached high enough accuracy (up to 98%) to prompt Tor to consider adopting defenses based on packet padding. Defenses such as *Walkie-Talkie* mainly remove features related to bursts of traffic without affecting packet timing. This was reasonable given that previous research on WF attacks ignored or deemphasized the use of packet timing information. In this thesis, we examine the extent to which packet timing can be used to facilitate WF attacks. In our experiment, we gained up to 61% accuracy on our unprotected dataset, 54% on our *WTF-PAD* dataset, and 43% on our *Walkie-Talkie* dataset using only timing-based features in an SVM classifier. Using a convolutional neural network (CNN), we got 88% accuracy on our unprotected dataset, and 76% and 47% accuracy on our *WTF-PAD* and *Walkie-Talkie* dataset respectively. We intend to investigate further to develop an effective and robust WF attack using packet timing.

Table of Contents

Acknowledgments iv

Abstract v

List of Figures viii

List of Tables x

Chapter 1. Introduction 1
- 1.1 Research Agenda . . . 3
- 1.2 Thesis Organization . . . 4

Chapter 2. Background 5
- 2.1 Tor Network . . . 5
- 2.2 WF Attack Model . . . 6
- 2.3 Machine Learning . . . 8
  - 2.3.1 k-Nearest Neighbor (k-NN) . . . 8
  - 2.3.2 Support Vector Machines (SVM) . . . 9
- 2.4 Deep Learning . . . 9
  - 2.4.1 Convolutional Neural Network (CNN) . . . 10

Chapter 3. Related Work 13
- 3.1 WF Attack using Traditional Machine Learning . . . 14
  - 3.1.1 WF Attack using $k-$NN . . . 14
  - 3.1.2 WF Attack using RDF . . . 15
  - 3.1.3 WF Attack using CUMUL . . . 15
- 3.2 Deep Learning in WF Attack . . . 16
  - 3.2.1 WF Attack by SDAE . . . 16
  - 3.2.2 Automated WF . . . 16

3.2.3 Deep Fingerprinting Attack . . . 16
3.3 WF Defenses . . . 17
3.3.1 WTF-PAD . . . 18
3.3.2 Walkie-Talkie . . . 18

Chapter 4. Packet Timing in Website Fingerprinting **19**
4.1 Data Collection . . . 19
4.1.1 Data Collection Process . . . 19
4.1.2 Datasets . . . 21
4.2 Experimental Design . . . 21
4.2.1 Feature Selection and Extraction . . . 22
4.2.1.1 Tuning the Number of Bins . . . 27
4.2.2 Hyperparameter Tuning in CNN . . . 28

Chapter 5. Evaluation & Results **31**
5.1 Experimental Environment . . . 31
5.2 Results . . . 32
5.2.1 RQ1: Packet Timing Features . . . 32
5.2.2 RQ2: Classification Value of Timing Features . . . 32
5.2.3 RQ3: WF Attack with Timing Features . . . 34
5.2.3.1 WF Attack with Defended Tor Traffic . . . 34
5.3 Discussion . . . 36

Chapter 6. Conclusion & Future Work **37**

References **38**

List of Figures

2.1 Tor Network. 5

2.2 Website Fingerprinting Attack. 6

2.3 A Visualization of Bursts – five outgoing bursts interspersed with five incoming bursts. 7

2.4 Architecture of CNN. 10

2.5 Operation of the convolution layer. 11

4.1 Median Packet Time (MED). 22

4.2 Variance of the Packet Times. 23

4.3 Time from the First Packet to the Last Packet (Interval). . . . 23

4.4 Interval between the medians of B_1 and B_2 (*IMD*). 24

4.5 Interval between the start of B_1 and the start of B_2. 24

4.6 Interval between the end of B_1 and the start of B_2 (IBD). . . 25

4.7 Interval between the start of B_1 and the start of B_2, where both B_1 and B_2 are incoming (to the client). 26

4.8 Interval between the start of B_1 and the start of B_2, where both B_1 and B_2 are outgoing (from the client). 26

4.9 Processing Features. 27

5.1 Undefended Tor Traffic: Attack Accuracy with CNN. 33

5.2 Defended Tor Traffic with WTF-PAD Defense: Attack Accuracy with CNN. 35
5.3 Defended Tor Traffic with W-T Defense: Attack Accuracy with CNN. 35

List of Tables

4.1 Datasets. 21
4.2 Tuning Number of Bins. 28
4.3 Hyperparameters Tuning of CNN Model for Undefended and WTF-PAD Dataset. 29
4.4 Hyperparameters Tuning of CNN Model for Walkie-Talkie Dataset. 30

5.1 *Undefended Tor*: Attack accuracy. 33
5.2 *Undefended Tor*: Attack accuracy in CNN. 33
5.3 *Defended Tor*: Attack accuracy. 34

Chapter 1

Introduction

We share a lot of our personal information either willingly or unwillingly on the Internet. Since the United States Congress has upheld the rights to ISP providers to sell our browsing history [8], concern for the privacy of this personal information is increasing day by day. Unfortunately, using a VPN or other basic protections cannot effectively help Internet users to protect their privacy [7]. For example,

Using an anonymity system such as Tor is a more effective solution to keep users' online activities anonymous while browsing the Internet. Tor is one of the most popular privacy-enhancing technologies, with more than two million users each day. Tor is, however, vulnerable to traffic analysis attacks. Traffic analysis enables an attacker analyze the network traffic stream to deduce information about the client's activities and communication. Website fingerprinting (WF) is one such attack, and it has received significant attention by both researchers [9, 19] and Tor developers [18].

In a WF attack, a passive local eavesdropper collects network traffic passing

between the client and entry node. From the collected traffic, the attacker then extracts various features and feeds them into a machine learning classifier trained to identify which website the client is visiting. Prior work has shown that this kind of attack is very effective, reaching over 90% accuracy [9, 15, 23, 19, 21].

The Tor Project has given much attention to building defenses against WF attacks [17]. The state-of-the-art attacks emphasize *bursts*, sequences of packets in a single direction. Because of this, defenses primarily seek to obscure burst patterns. This still leaves the timing of packets as a largely untapped and unprotected resource for features for WF attacks.

Prior work on WF attacks discounted timing information [9]. This is because timing characteristics change on each visit to the site, which makes it hard to extract consistent patterns. In this thesis, we investigate a novel WF attack using packet timing information. Since individual packet times are generally unreliable as features, we identify more robust timing-related features from the data. We create new representations of timing information that capture meaningful features in a reliable way.

We show the effectiveness of our features using traditional machine learning and deep learning algorithms. Using our new timing features with k-nearest neighbor (k-NN) and support vector machine (SVM) classifiers, we get 51% and 61% attack accuracy, respectively. Using the CNN deep-learning classifier, we get 88% accuracy. These preliminary results indicate that timing

information is useful for WF attack, and it is possible to develop an effective WF attack using timing information. Additionally, our timing features show higher accuracy than that of using packet direction features against *Walkie-Talkie* (W-T) defense as well.

1.1 Research Agenda

From our literature review, we find some limitations in the prior work on WF. Prior work on attacks deemphasized or ignored the value of packet timing information. This is because timing information varies from instance to instance, which makes it hard to find any consistent pattern to develop an attack. In this thesis, we have used packet timing information for WF attack development. As we cannot use raw timing information because of the larger variability between instances, we need to find a new representation of the timing information that we can feed into the machine learning classifier. The research questions (RQ) we investigated in this research are as follows:

RQ1: How can we represent packet timing information in a way that robustly reveals patterns over different instances of a site?

RQ2: How useful are those new representations for developing WF attack?

RQ3: Do the new representations also provide classification value when WF defenses are in place?

1.2 Thesis Organization

This thesis is organized as follows: Chapter 2 provides background of Tor, website fingerprinting, machine learning, and deep learning. Chapter 3 presents related work of both website fingerprinting attacks and defenses. The motivation for using packet timing information, the way we select timing features, and the feature extraction process are discussed in Chapter 4. We explain our evaluation of the experiments and the results in Chapter 5. We then conclude this thesis and discuss future work in Chapter 6.

Chapter 2

Background

In this chapter, first we briefly describe the Tor network in Section 2.1. We provide background on the website fingerprinting (WF) attack model in Section 2.2. In Section 2.3 and Section 2.4, we provide background on machine learning and deep learning respectively, as they are used in WF attacks.

2.1 Tor Network

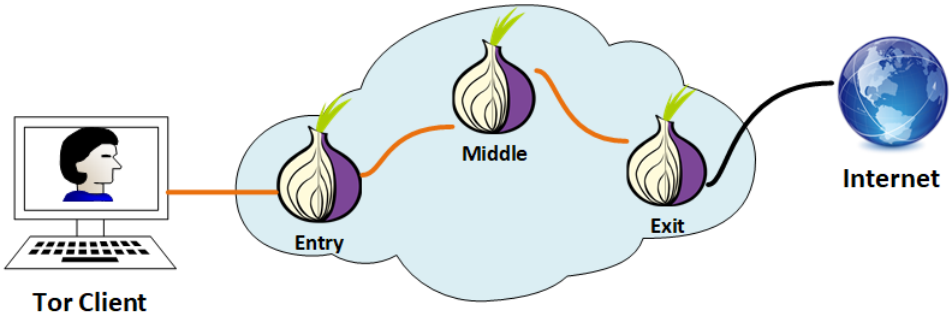


Figure 2.1: Tor Network.

Tor keeps the user activities anonymous to ensure privacy of the Tor client. At the time of browsing the Internet on Tor, the user can protect her privacy

by separating her online activities from her identity. Tor is one of the most popular low-latency anonymity systems with more than two million users each day. Tor ensures the anonymity of the users' activities with the help of three nodes that constitute a Tor circuit: *entry or guard* node, *middle* node, and *exit* (see Figure 2.1). For example, the network traffic from the Tor client first goes to the guard node, then guard node passes those traffic to the middle node, then middle node to the exit node, and finally exit node to the web server. Each node is only aware of the previous and next node in the circuit. For example, the guard only knows about the identity of the client and the middle, while the exit only knows about middle and the destination of the client.

2.2 WF Attack Model

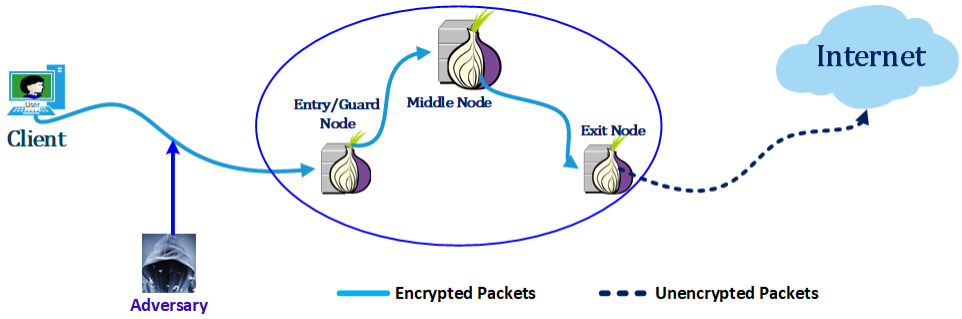


Figure 2.2: Website Fingerprinting Attack.

WF attack enables a local passive eavesdropper to identify the destination of a Tor client's connection by analyzing the network traffic passing from the client to the guard (see Figure 2.2). The term *local* means that the attacker is sitting

locally between the client and the guard node, and thus he knows the identity of the client. For example, the client and the attacker are in same wireless network, and the attacker has the capacity to sniff the wireless traffic. The term *passive* means that the attacker does not modify, insert, or delete any packets', rather, she just observes the traffic. In this threat model, we assume that an attacker has the capacity to monitor the traffic between a client and guard node, and/or she controls the guard node.

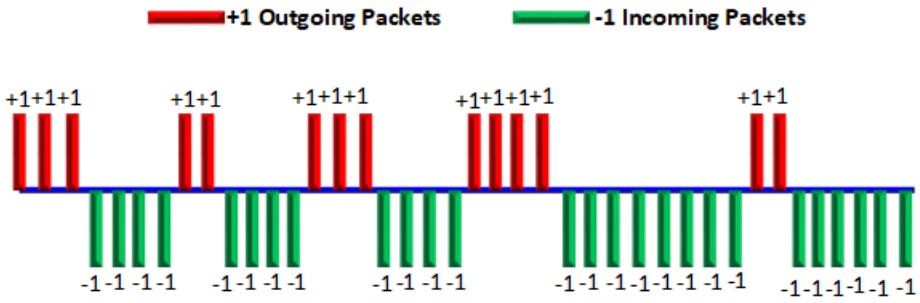


Figure 2.3: A Visualization of Bursts − five outgoing bursts interspersed with five incoming bursts.

A WF attacker can collect traffic traces from several websites of her interest such as `cnn.com`, `twitter.com`, and so on. These websites are called *monitored* websites. She can extract statistical data as features from the collected network traffic, such as the total number of incoming packets, the total number of outgoing packets, the timing of each packet, and so on. A particularly important class of features is related to *bursts* sequences of consecutive incoming and outgoing packets (see Figure 2.3). She can use the features to train a machine learning classifier such as k-nearest neighbors (k-NN) and

support vector machines (SVM). Using the trained classifier, she can identify the client's destination websites.

2.3 Machine Learning

In our experiments, we employ two widely used machine learning (ML) techniques: k-nearest neighbors (k-NN) and support vector machines (SVM). In the following sections, we provide brief explanation of this two techniques.

2.3.1 k-Nearest Neighbor (k-NN)

The k-NN algorithm can be used for both classification and regression. In website fingerprinting (WF), we use k-NN as a classifier. As k-NN is a supervised ML algorithm, we feed the data along with the associated label to train the classifier. In our case, the instances of the websites are the data, and the name of the websites are the labels.

We train the k-NN classifier with training examples (x, y) where x is the feature and y is the target label or class. The objective is to learn the function $g : X \to Y$ to find the relationship between x and y so that $g(x)$ can predict the correct label of an unseen x. For every data points, the algorithm measures a similarity index between the k nearest neighbors and the data point x. A popular similarity matrix among researchers is Euclidean distance.

2.3.2 Support Vector Machines (SVM)

Support vector machines (SVM) is one of the popular traditional machine learning techniques for solving classification problems. The basic idea is to plot the data points (features) in a n-dimensional space, where n is the number of features. weight or value of each feature is the coordinate value in the space. The Classification is done by separating the features by finding hyper-planes that divide data points in each class [26].

We want our features to be as farther as possible from the hyper-planes. Hyper-planes can easily be useful to classify the linear features. Classification becomes challenging when the data points are non-linear. For example, network traffic data used in WF attacks have non-linear features. To address this, we apply a technique called *kernel trick*. The kernel trick transforms low-dimensional data into high-dimensional data. In the higher dimension, the data points become linearly separable [14].

2.4 Deep Learning

Deep learning is a type of machine learning that uses neural network method to train the model to classify more accurately than the traditional machine learning algorithm such as k-NN and SVM. There are different models of deep learning such as stacked denoising autoencoder (SDAE), long short-term memory (LSTM), and convolutional neural network (CNN), and so on. In our experiment, we mainly use convolutional neural network (CNN) as it shows

better performance than SDAE and LSTM [3, 19, 21]. Hence, we are going to give a brief explanation of CNN.

2.4.1 Convolutional Neural Network (CNN)

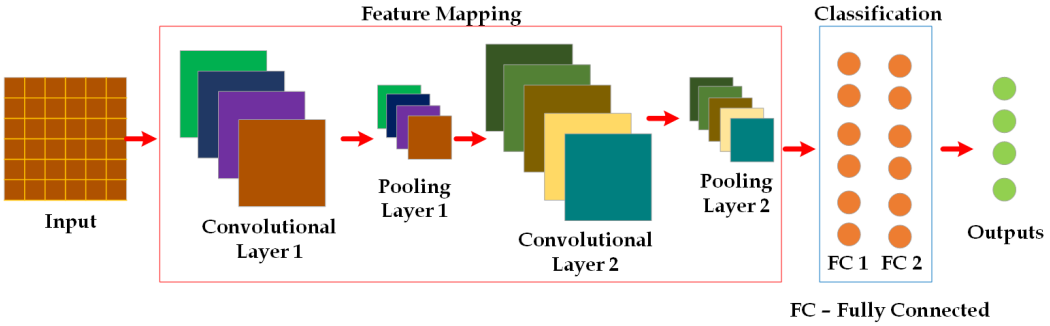


Figure 2.4: Architecture of CNN.

The convolutional neural network (ConvNet or CNN) has been proven to be a very powerful deep learning model in the area of computer vision to classify images. There are four major operations of a CNN model. We can see a visualization of a CNN architecture [1] from Figure 2.4.

- Convolution
- Activation Function
- Sub-sampling or pooling
- Classification or fully connected

[1] Adapted from Saleh and Ausif [4].

Convolution Layer

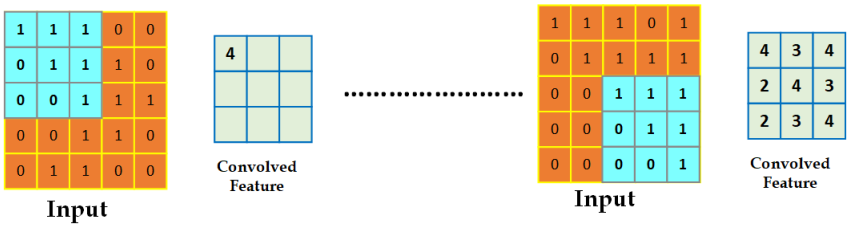


Figure 2.5: Operation of the convolution layer.

We feed our data points or raw features into a matrix. We map features from the original feature matrix by convolution. For example, as shown in Figure 2.5 the original input matrix is 5x5, we convolute by 3x3 matrix [2]. This 3x3 matrix is called the filter of the convolution. The convolution is performed by moving the 3x3 matrix by 1 pixel over the 5x5 matrix and taking the multiplication value of the 3x3 matrix. Moving by 1 pixel is called *stride* 1. If we move by 2 pixel, then the *stride* would be 2. We get the convolved features by scanning the whole image by this 3x3 matrix.

Activation Function

Activation function is used to introduce non-linear property in a neural network. This function converts the input values to the output values. For example, before getting the final convolved features in a CNN, we apply activation function to introduce non-linearity into the convolved features. It is a very important part of a CNN as a non-linear operation in a otherwise linear system.

[2] Adapted from [2]

The activation function should be selected based on the type of dataset− an activation function that works for image classification may not be suitable for WF attacks.

There are several widely used activation functions such as sigmoid, rectified linear unit (ReLU), exponential linear unit (ELU), and hyperbolic tangent (tanh). In our CNN models, we use the ReLU, ELU, and tanh activation functions.

Sub-sampling or Pooling Layer

In this layer, we sub-sample or down-sample the convolved features. This layer reduces the dimensions of each mapped feature. But it keeps the most important features. There are some variations of pooling: max, average, sum, and so on. In our model, we use max pooling. We do that by taking the maximum of the convolved feature.

Fully Connected Layer

The classification of a CNN is done by the fully connected (FC) layer. Every neuron is connected to each other in the fully connected layer. The convolutional and pooling layers give us high-level features. FC layer does the actual classification by softmax regression. Softmax is a regression model for multi-class classification. The sum of the probabilities of this layer is one.

Chapter 3

Related Work

Tor is the most popular anonymity systems to protect user privacy online. However, an attacker can deanonymize the activity of a Tor client by the website fingerprinting (WF) attack. Indeed, to combat this attack, Tor has deployed WF defenses [17], though these are not considered effective.

In prior works, WF attacks and defenses are investigated in two settings: close-world (CW) and open-world (OW) [5, 6, 9, 10, 23, 24]. The assumption of the close world is that the Tor client is visiting one or more websites from a small set of websites. Only in OW setting, the attacker trains his classifier with only these websites. On the other hand, in an OW setting, the client is assumed to visit any website including the monitored websites. But it is not realistically possible for an attacker to train his classifier with all the possible websites. So, the attacker trains the classifier with a set of sites only the monitored website. The state-of-the-art WF attack reaches up to 98% accuracy in the closed-world setting, and it achieves 95.7% true positive rate (TPR) with 0.7% of false positive rate (FPR) in the open-world setting.

In this chapter, we discuss WF attacks using traditional machine learning and using deep learning in more detail in Section 3.1 and Section 3.2, respectively. We discuss WF defenses in Section 3.3.

3.1 WF Attack using Traditional Machine Learning

In 2009, Herrmann et al. [10] published a WF attack using IP packet size for their classifier, but it does not work on Tor, which has fixed-sized packets. Panchenko et al. designed a new attack adding more features: packet volume, packet direction, and timing [16]. They used support vector machines (SVM) for classification and achieved 55% accuracy. In 2012, SVM was again used by Cai et al. who proposed a new attack based on a new representation of the classification instances [5]. Their SVM was based on the Damerau-Levenshtein edit distance and using SVM kernel trick to pre-compute distances between the traces. This same attack was improved by Wang and Goldberg [24] achieving 91% accuracy. In the rest of the section, we are going to give a brief overview of three more effective and efficient WF attacks.

3.1.1 WF Attack using $k-$NN

In 2014, Wang et al. proposed a new attack using a k-nearest neighbor (k-NN) classifier on a large feature set with weight adjustment [23]. They modified the typical k-NN classifier with a weighted distance function. In the general k-NN, Euclidean distance is usually used. There are two phases of the attack: the weight-learning phase and the classification phase. The weights that are

learned in the first phase are used for classification in the second phase. This attack was the first to use a diverse set of features (bursts, packet ordering, concentration of the packets, number of incoming and outgoing packets, and so on) from the traffic information in a WF attack. In a closed-world setting of 100 websites, they achieved over 90% accuracy.

3.1.2 WF Attack using RDF

Hayes et al. use a novel feature extraction and selection method: they use random decision forests (RDF) to rank features [9]. The classification is performed by the K-NN classifier using the ranked features of the RDF. This attack also achieved over 90% accuracy in close-world scenario.

3.1.3 WF Attack using CUMUL

In 2016, Panchenko et al. proposed a new attack improving features based on packet size, packet ordering, and packet direction [15]. Their attack uses SVM to classify the websites. Their significant contribution is their new feature set. In addition, they collected a new data set that is more realistic. Most work collected the traffic of the 100 most popular websites listed by Alexa known as the *Alexa Top 100* [1]. Panchenko et al. collected their data based on trends in Twitter, Google, and random Google search results of web pages. They also collected data from websites censored in China.They were able to achieve 92% accuracy in close-world scenario.

3.2 Deep Learning in WF Attack

3.2.1 WF Attack by SDAE

Abe and Goto is the first to explore the effectiveness of deep learning (DL) effectiveness in traffic analysis [3]. They used a *Stacked Denoising Autoencoder* (SDAE) model as their deep learning model, with a simple input data representation based on incoming and outgoing packet traces. They got 88% accuracy using deep learning without any manual selection of packet features. They used small datasets, which is a reason for their lower attack accuracy. However, they only considered one type of data representation that completely omit timing of the packets.

3.2.2 Automated WF

Rimmer et al. [19] investigated automated feature engineering in WF attacks. They studied three deep learning models: *Stacked Denoising Autoencoder (SDAE)*, *Convolutional Neural Network (CNN)*, and *Long-Short Term Memory (LSTM)*. They also collected large datasets suitable for deep learning models, with 900 websites and 2500 traces for each site. They achieved 96% accuracy in a closed-world setting.

3.2.3 Deep Fingerprinting Attack

Sirinam et al. [21] extensively investigated the use of deep learning in website fingerprinting. Their attack could outperform all the previous attack accu-

racy reaching over 98% attack accuracy. They evaluated their deep learning model by 100 classes containing 1000 traces each. They developed a powerful convolutional neural network (CNN) model for their attack. Their attack achieved up to 90% accuracy against *WTF-PAD*, one of the state-of-the-art defenses. They showed the effectiveness of their attack even though the defense is in place. Their attack also achieved 49.7% accuracy against *Walkie-Talkie* defense.

3.3 WF Defenses

WF defenses aim to reduce the effectiveness of these attacks. For defenses against WF attacks, we are interested in the state-of-the-art defenses that are claimed to be highly efficient and have low overheads in latency and bandwidth for users. Tor implemented a HTTP pipeline based defense [17] to combat WF attacks. The purpose of this defense is to change the order of the requests when the number of requests exceeds the depth of the pipeline. This objective is achieved by randomizing the maximum number of requests in a pipeline. The bandwidth overhead of this defense is zero. Recently, Tor has updated this defense mechanism [18] because of the emergence of newly developed attacks. However, neither version of Tor's defenses could reduce the attack accuracy of newly developed attacks [3, 5, 19, 21, 23, 24].

In the next section, we are going to discuss more about the two state-of-the-art defenses, *WTF-PAD* [13] and *Walkie-Talkie* [25].

3.3.1 WTF-PAD

Website Traffic Fingerprinting Protection with Adaptive Defense (WTF-PAD) [13] applies the adaptive padding technique to WF defense in Tor [20]. Adaptive padding works by sending data packets with padding of a certain distribution and with no delay. By doing so, adaptive padding does not incur any latency overhead. However, it does incur 54% bandwidth overhead.

3.3.2 Walkie-Talkie

Wang and Goldberg [25] developed *Walkie-Talkie* (W-T) defense. Their defense is based on the concept of half-duplex communication and burst molding. The default communication mechanism of Tor is full duplex. In full duplex communication mode, we simultaneously send outgoing packets and receive incoming packets. W-T defense requires the communication of Tor browser to be half-duplex. In half-duplex communication, we send request of packets and wait to receive all the requested packets, after receiving all the packets, we send another request of packets. The client and the guard have to use half-duplex communication mode to implement this defense. This half duplex mode forms a sequence of outgoing and incoming bursts. The client molds the bursts of the traces of sensitive website and non-sensitive websites together. The idea is to make two websites look exactly the same to the attacker. Their defense has 31% bandwidth overhead and 34% latency overhead.

Chapter 4

Packet Timing in Website Fingerprinting

In this chapter, we discuss the motivation for using packet timing information, the way we select timing features, and the feature extraction process. we discuss data collection process in Section 4.1.1 and our datasets in Section 4.1.2. We discuss our experimental design in Section 4.2. The process of feature selection and extraction is discussed in Section 4.2.1. We discuss hyperparameters tuning of our CNN models in Section 4.2.2.

4.1 Data Collection

4.1.1 Data Collection Process

As we have used the datasets provided by [21], we are going to provide a brief overview of the data collection process that Sirinam et al. used. Firstly, the top Alexa 100 websites are selected as the monitored sites. Homepage of each site is visited 1,250 times and the generated traffic of each visit is dumped in terms of each visit using `tcpdump`. We call each visit of a site as an instance

of that site.

The data collection is done based on the batch method provided by Wang and Goldberg. The batch method is useful for managing long and short term time variance. A batch contains 25 visits of a site at a time. After that, the crawler visits another site 25 times and this process continues. Batching helps to collect the traffic over time that helps to avoid the IP to be banned. In addition, it helps to capture traffic with *variation*.

The websites are visited using `tor-browser-crawler` [12]. This crawler is more realistic in the sense that it helps to emulate the real behavior of a Tor client. Hence, this crawler is more realistic to collect Tor traffic than the traditional crawling tools such as `wget` or `curl` for our purpose as it has the necessary settings for that.

At the end of crawling, the sanity check of the datasets is performed. At first, the instances that do not have more than 50 packets and that do not contain any incoming and outgoing packets are discarded because they do not contain enough information for WF attack. After this, 95 valid sites and 1000 instances of each site are selected as the monitored undefended datasets. For the *WTF-PAD* defense, the defense is applied to the datasets in simulator to make defended monitored datasets.

4.1.2 Datasets

We use three datasets in our experiments, as shown in Table-4.1. These datasets have been used for the previous work [21, 25] in WF research. We use both defended and undefended Tor traffic. *Defended* Tor traffic means that a defense mechanism such as *WTF-PAD* or *Walkie-Talkie* (W-T) is applied to those traffic. *Undefended* Tor traffic means the traffic with the default settings of Tor. For undefended Tor traffic and Tor traffic defended with *WTF-PAD* [13], we use the dataset generated by Sirinam et at. [21]. The *Walkie-Talkie* defense requires changes to the underlying browser. At the time we performed our experiments, the dataset from Sirinam et al. did not include W-T data. We thus use the smaller dataset used by Wang et al. [25].

Table 4.1: Datasets.

Dataset	**Source**	**Classes**	**Instances in each class**	**Total**
Undefended	Sirinam et al.	95	1000	95,000
WTF-PAD	Sirinam et al.	95	1000	95,000
Walkie-Talkie	Wang et al.	100	100	10,000

4.2 Experimental Design

In this section, we explain our process of selecting and extracting timing features as well as the process of tuning hyper-parameters for our CNN models. We have used the K-NN model of Wang et al. [23] and SVM model of

Panchenko et al. [15] to best compare our results. Using non-timing based features, they achieve over 90% accuracy. Our intuition was to compare the results using only packet timing information.

4.2.1 Feature Selection and Extraction

We developed several timing-based features that would be more robust from instance to instance of each website than relying on specific packet timings. Since prior work on attacks relies heavily on bursts, we base our timing features on burst-level characteristics. Three of our features are focused on the timing of packets inside a single burst.

Median Packet Time (MED):

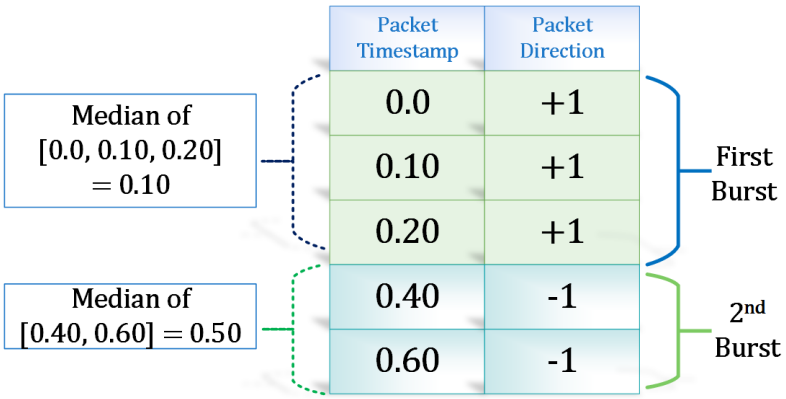


Figure 4.1: Median Packet Time (MED).

We can see the extraction process of median packet time within a burst from Figure 4.1. For example, we take the time sequence of the first burst: [0.0, 0.10, 0.20], and take the median of these three values.

Variance of the Packet Times:

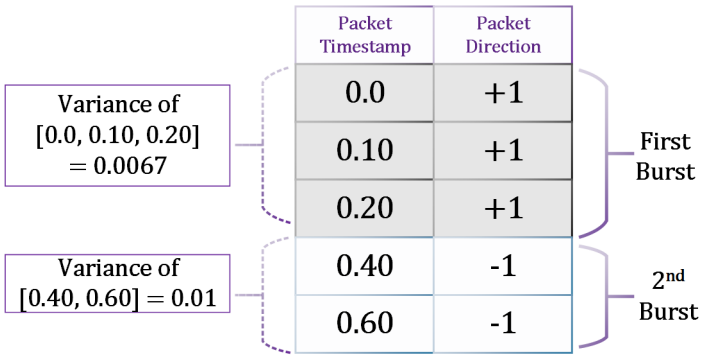


Figure 4.2: Variance of the Packet Times.

We can see the extraction process of variance packet time within a burst from Figure 4.2. For example, we take the time sequence of the first burst: [0.0, 0.10, 0.20], and take the variance of these three values.

Time from the First Packet to the Last Packet (Interval):

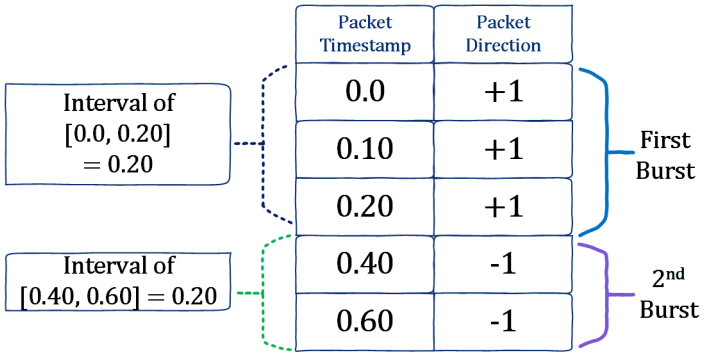


Figure 4.3: Time from the First Packet to the Last Packet (Interval).

We can see the extraction process of interval of the packet times within a burst from Figure 4.3. For example, we take the time sequence of the first burst: [0.0, 0.20], and take the interval of these two values.

The other five features consider two consecutive bursts B_1 and B_2.

Interval between the medians of B_1 and B_2 (IMD):

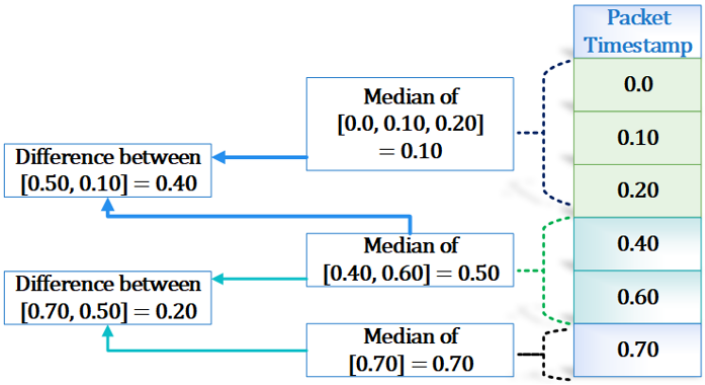


Figure 4.4: Interval between the medians of B_1 and B_2 (*IMD*).

We can see the extraction process of the interval between the medians of B_1 and B_2 (IMD) from Figure 4.4. For example, we take the median of the burst one and the burst two: [0.10, 0.50], and take the interval of these two values.

Interval between the Start of B_1 and the Start of B_2:

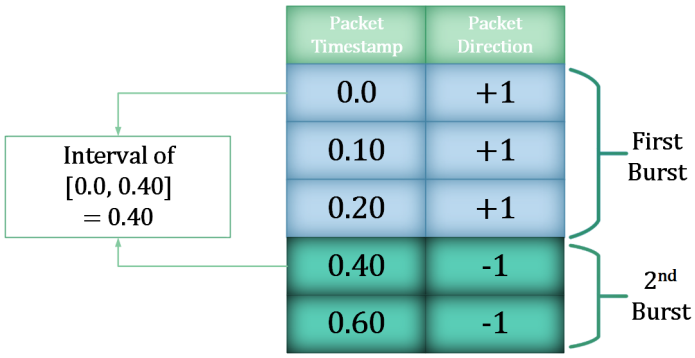


Figure 4.5: Interval between the start of B_1 and the start of B_2.

We can see the extraction process of the interval between the start of B_1 and

the start of B_2 from Figure 4.5. For example, we take the time of the first packet of the burst one and the time of the first packet of burst two: [0.00, 0.40], and take the interval of these two values.

Interval between the End of B_1 and the Start of B_2 (IBD):

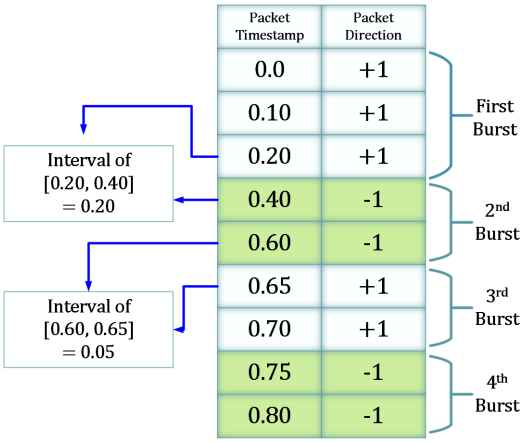


Figure 4.6: Interval between the end of B_1 and the start of B_2 (IBD).

We can see the extraction process of the interval between the end of B_1 and the start of B_2 (IBD) from Figure 4.6. For example, we take the time of the last packet of the burst one and the time of the second packet of burst two: [0.20, 0.40], and take the interval of these two values.

Interval between the Start of B_1 and the Start of B_2, where both B_1 and B_2 are Incoming (to the client):

We can see the extraction process of the interval between the start of B_1 and the start of B_2, where both B_1 and B_2 are incoming (to the client) from Figure 4.7. For example, we take the time of the first packet of the first incoming

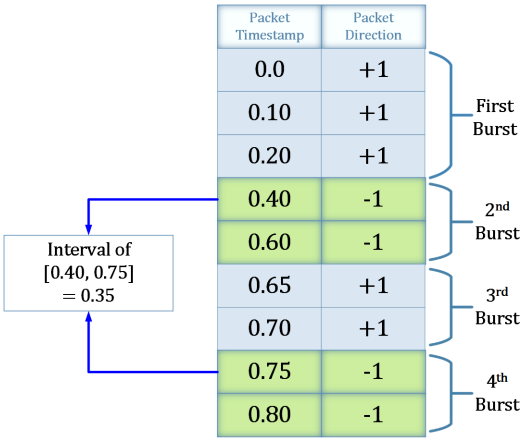


Figure 4.7: Interval between the start of B_1 and the start of B_2, where both B_1 and B_2 are incoming (to the client).

burst and the time of the first packet of the second incoming burst: [0.40, 0.75], and take the interval of these two values.

Interval between the Start of B_1 and the Start of B_2, where both B_1 and B_2 are Outgoing (from the client):

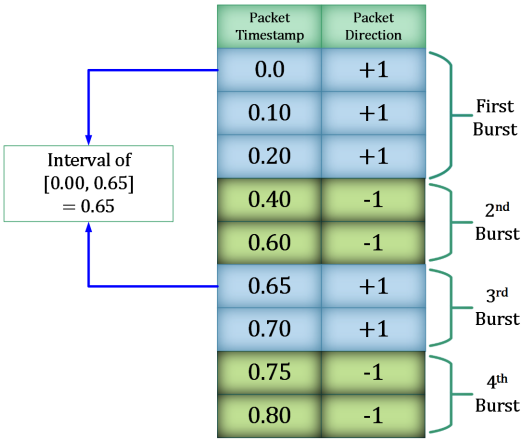


Figure 4.8: Interval between the start of B_1 and the start of B_2, where both B_1 and B_2 are outgoing (from the client).

We can see the extraction process of the interval between the start of B_1 and the start of B_2, where both B_1 and B_2 are outgoing (from the client) from Figure 4.8. For example, we take the time of the first packet of the first outgoing burst and the time of the first packet of the second outgoing burst: [0.00, 0.65], and take the interval of these two values.

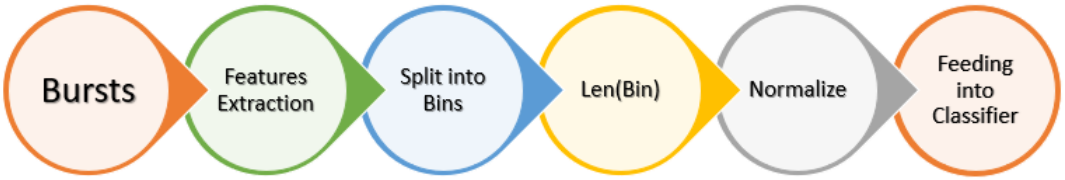


Figure 4.9: Processing Features.

To create features that would be robust to different instances, we further process the extracted features (see Figure 4.9). Using the data from all the instances over all websites, we create a histogram of b equal-sized bins for each feature. Then, for each instance, we extract the features from raw data, put them into their respective bins, and take the length of each bin. Finally, we normalize the length of each bin and feed those normalized values into the classifiers.

4.2.1.1 Tuning the Number of Bins

We tune the number of bins for each dataset. We started with $b = 20$. We explore the variation in accuracy with the change of the bin sizes. We think that number of features a bin can contain impact the classification accuracy.

For example, if we increase the number of bins, every bin will contain less features but are more fine-grained. On the other hand, if we decrease the number of bins, every bin will contain more features but are less fine-grained. So we have to make a trade-off between the number of bins and the number of features a bin contain. Taking that into account, we explore different number of bins to find the best one based on the accuracy. We present our search space of bin size in Table 4.2. We could not go beyond $b = 25$ for *Walkie-Talkie* defense because the highest number of packets of that dataset is less than 200. Hence, the more we increase bin size, the lesser information the bin contain. We finally select $b = 20$ for undefended and *WTF-PAD* defense and $b = 10$ for *Walkie-Talkie* defense.

Table 4.2: Tuning Number of Bins.

Dataset	**Number of Bins Search**	**Final Bin Size**
Undefended	5, 10, 15, 20, 25, 30, 35, 40, 45, 50	20
WTF-PAD	5, 10, 15, 20, 25, 30, 35, 40, 45, 50	20
Walkie-Talkie	5, 10, 15, 20, 25	10

4.2.2 Hyperparameter Tuning in CNN

Different types of datasets require different hyper-parameters in the CNN model to make the model robust. Usually deep learning models require a lot of data for training. We have sufficient data to use a deep CNN model in the undefended and *WTF-PAD* datasets. However, for the *Walkie-Talkie* de-

fense we have a relatively small dataset. This means we need to use a shallow CNN model to avoid overfitting. We provide an overview of the parameters of our models in Table 4.3 and Table 4.4.

Table 4.3: Hyperparameters Tuning of CNN Model for Undefended and WTF-PAD Dataset.

		Final Parameters	
Hyperparameters	**Search Range**	**Undefended**	***WTF-PAD***
Input Dimension	[15 ... 400]	160	160
Optimization Function	[Adam, Adamax, SGD]	Adamax	Adamax
Number of Filters			
Layer 1 [Conv1, Conv2]	[8 ... 64]	[32, 32]	[32, 32]
Layer 2 [Conv3, Conv4]	[32 ... 128]	[64, 64]	[64, 64]
Layer 3 [Conv5, Conv6]	[64 ... 256]	[128, 128]	[128, 128]
Layer 4 [Conv7, Conv8]	[128 ... 512]	[256, 256]	[256, 256]
Filter Sizes	[3 ... 10]	[8, 8, 8, 8]	[8, 8, 8, 8]
Pooling	Max	Max	Max
Learning Rate	[0.001 ... 0.05]	0.001	0.001
Training Epochs	[30 ... 500]	50	100
Mini-batch Size	[16 ... 256]	128	128
Activation Functions	[Sigmoid, Tanh, ReLU, ELU]	ELU, ReLU	ReLU, Tanh
Number of FC Layers	[1 ... 4]	2	2
Hidden units (each FCs)	[128 ... 512]	[512, 512]	[512, 512]
Dropout [Pooling, FC1, FC2]	[0.1 .. 0.8]	[0.1, 0.7, 0.5]	[0.1, 0.5, 0.5]

To further avoid overfitting, we add a kernel regularizer, dropout [22], and batch-normalization in our models. For the kernel regularizer, we adopt l_2 regularization, also called weight decay. Kernel regularization is usually implemented in each convolutional layer. Dropout is implemented in the fully connected (FC) layers. Dropout randomly disconnects some of the units in the

Table 4.4: Hyperparameters Tuning of CNN Model for Walkie-Talkie Dataset.

Hyperparameters	**Search Range**	**Final Parameters** ***Walkie-Talkie***
Input Dimension	[15 ... 200]	80
Optimization Function	[Adam, Adamax, SGD]	Adamax
Number of Filters		
Layer 1 [Conv1, Conv2]	[8 ... 64]	[16, 16]
Layer 2 [Conv3, Conv4]	[32 ... 128]	[32, 32]
Layer 3 [Conv5, Conv6]	[64 ... 256]	[64, 64]
Filter Sizes	[3 ... 10]	[8, 8, 5, 5]
Pooling	Max	Max
Learning Rate	[0.001 ... 0.05]	0.001
Training Epochs	[30 ... 500]	100
Mini-batch Size	[16 ... 256]	64
Activation Functions	[Sigmoid, Tanh, ReLU, ELU]	ReLU, Tanh
Number of FC Layers	[1 ... 4]	1
Hidden units (each FCs)	[128 ... 512]	512
Dropout [FC1]	[0.1 .. 0.6]	0.5

FC layers and make the model more robust for testing. We add batch normalization [11] in both the convolutional layer and the FC layer. We use dropout in both of our CNN models. We use kernel regularizer in the CNN model for the *Walkie-Talkie* dataset and batch normalization in the CNN model for the undefended and *WTD-PAD* datasets.

Chapter 5

Evaluation & Results

In this chapter, we discuss the evaluation of our timing features and present results based on the three research questions we investigated. First, we provide a brief overview of our experimental environment in Section 5.1. Secondly, we address our three research questions and the experimental results in Section 5.2. Then we discuss our results in Section 5.3.

5.1 Experimental Environment

We use a PC with a Core i7-6700 processor with 8 cores and 32GB RAM, plus a NVIDIA Quadro K1200 GPU with 4GB GPU memory. For compiling deep learning models to run on a GPU, we use Cuda Version 7.5. We develop our model in Python using the Keras deep learning framework as the front end and the Tensorflow deep learning framework as the back end.

5.2 Results

5.2.1 RQ1: Packet Timing Features

> **RQ1:** How can we represent packet timing information in a way that robustly reveals patterns over different instances of a site?

To respond this question, we selected and extracted eight timing features in a five steps process (see Figure 4.9). We extracted our timing features from the burst-level characteristics. Among the eight features, three features are based on the timing of a single burst. The other five features are based on the two consecutive bursts B_1 and B_2. Refer to Section 4.2.1 for the details of our timing features selection and extraction process.

5.2.2 RQ2: Classification Value of Timing Features

> **RQ2:** How useful are those new representations for developing WF attack?

We explored the use of all eight features, both separately and together, in both k-NN, and SVM for different settings of the number of bins b in each histogram. Based on our initial findings, we found that the most effective combination was the three features MED, IMD, and IBD with $b = 20$ bins, so we present results with these features.

In our initial phase experiments, we investigated and experimented our timing features with $K-$NN and SVM classifiers. As shown in Table-5.1, MED, IMD, and IBD all provide some classification value on their own, where random

Table 5.1: *Undefended Tor*: Attack accuracy.

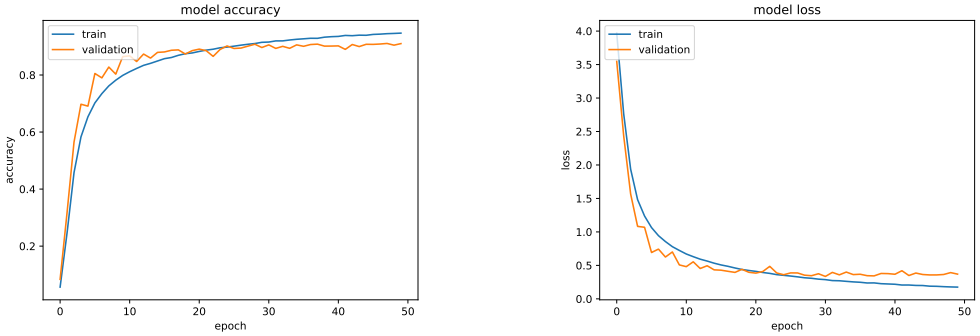
	Features			
Classifier	MED	IMD	IBD	**Combined**
k-NN	41%	20%	18%	51%
SVM	56%	24%	29%	**61%**

guessing would only reach 1% accuracy in this multi-class scenario. Together, these three features can get 61% accuracy with SVM on undefended Tor traffic.

Based on the findings of the initial experiments, we extended our experiments on timing features to the deep learning model. We adopted one of the most powerful deep learning classifiers: Convolutional Neural Network (CNN).

Table 5.2: *Undefended Tor*: Attack accuracy in CNN.

Classifier	**Combined Features**
CNN	**88%**



(a) Model Accuracy (b) Model Loss

Figure 5.1: Undefended Tor Traffic: Attack Accuracy with CNN.

Using the CNN classifier, we attain 88% testing accuracy (see Table- 5.2) with

undefended Tor traffic. We attain up to 98% of training accuracy and up to 92% of validation accuracy at the 50th epoch (see Figure-5.1). We performed our experiments with different structures of CNN model to understand the variation of our results. Finally, we selected the model that gave us the best accuracy. Refer to Table 4.3 for an overview of the search space of our model. While our results are not as good as the state-of-the-art classifiers, it is an encouraging result given that we are not using any of the features about bursts cited in prior work as being important for effective classification.

5.2.3 RQ3: WF Attack with Timing Features

RQ3: Do the new representations also provide classification value when WF defenses are in place?

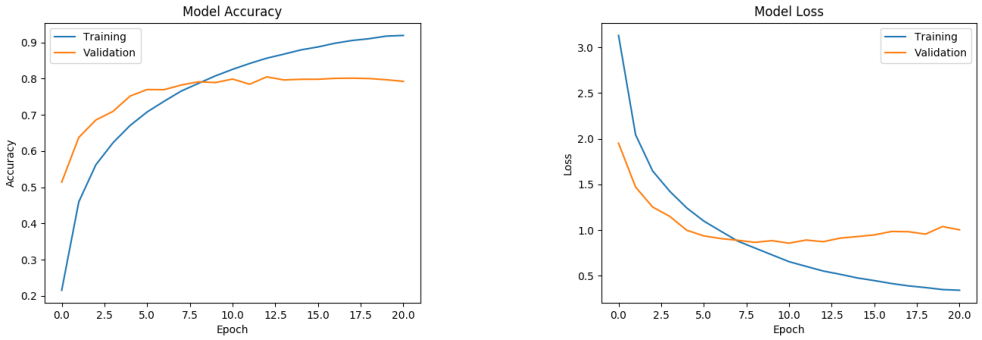
5.2.3.1 WF Attack with Defended Tor Traffic

Table 5.3: *Defended Tor*: Attack accuracy.

Classifier	WTF-PAD	W-T
SVM	54%	43%
CNN	**76%**	**47%**

We experimented with the combined feature set against the two defended datasets, using WTF-PAD and Walkie-Talkie (W-T), as reported in Table-5.3.

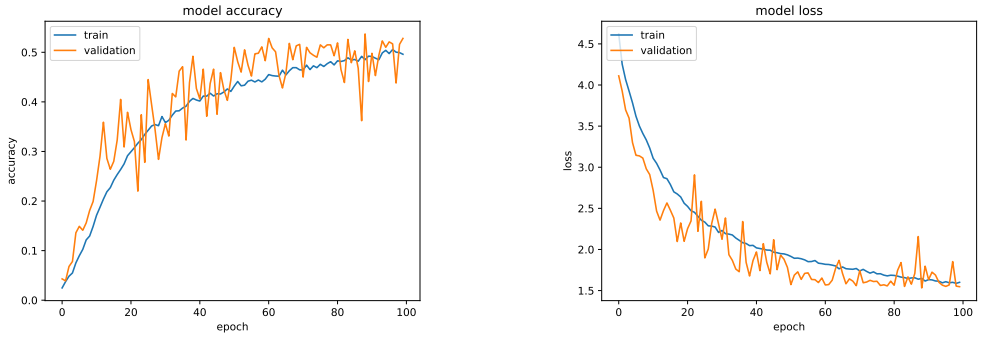
Against WTF-PAD, we attain 54% accuracy using SVM and 76% accuracy



(a) Model Accuracy (b) Model Loss

Figure 5.2: Defended Tor Traffic with WTF-PAD Defense: Attack Accuracy with CNN.

using CNN. Our CNN model gave us up to 96% of training accuracy and 78% of validation accuracy (see Figure- 5.2).



(a) Model Accuracy (b) Model Loss

Figure 5.3: Defended Tor Traffic with W-T Defense: Attack Accuracy with CNN.

Against W-T, we get 43% accuracy using SVM and 47% accuracy using CNN. Note that 47% is higher than any accuracy results reported on W-T to date

and approaching the 50% theoretical maximum accuracy claimed for W-T [25]. That maximum accuracy depends on not using timing data, so our approach might exceed 50% accuracy when both timing and burst data are considered.

5.3 Discussion

Our findings show that packet timing information can be an important candidate for feature for the WF attack. Using timing features provides reasonable attack accuracy against the state-of-the-art defenses as well. However, processing the timing features is a cumbersome task. First, it involves multiple steps to extract the fine-grained timing features. Secondly, tuning the number of bins is also challenging because it requires a trail and error process. No one specific number of bins can work for every dataset. The whole process requires multiple iterations from the start to the end. Another challenging task is hyperparameter tuning to find the best CNN model for our datasets. Finally, we have two datasets that are large enough for our deep learning models, but Walkie-Talkie (W-T) dataset is not large enough. A larger W-T dataset could give us better understanding of the accuracy using our timing features. However, we still get higher accuracy than the accuracies for the other attacks shown in the W-T paper [25] using only timing features.

Chapter 6

Conclusion & Future Work

In this thesis, we propose a novel WF attack based on extracting features from packet timing information. The main purpose of our research is to find useful timing features that can be used in WF attacks. We selected and extracted eight new timing features from packet timestamps in a five-step process. We show the effectiveness of our features in both traditional ML and deep learning models. We find that our features are robust over multiple noisy instances and provide useful classification value against both undefended and defended Tor traffic. Using timing features in our CNN model, we achieve 88% accuracy. We achieve 47% accuracy against W-T defense as well. In future work, we will investigate more to improve this attack and improve our model as well. We will investigate the performance of timing and direction features together. We plan to collect a larger W-T dataset to evaluate further the effectiveness of our timing features. We will also expand our evaluation to Onion Services and the open-world setting. In the future, our goal is to build a defense against this attack, as state-of-the-art defenses do not focus on obscuring the pattern of timing information.

References

[1] "Alexa," http://www.alexa.com.

[2] "Feature extraction using convolution," http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution.

[3] K. Abe and S. Goto, "Fingerprinting attack on Tor anonymity using deep learning," *Proceedings of the Asia-Pacific Advanced Network*, vol. 42, pp. 15–20, 2016.

[4] S. Albelwi and A. Mahmood, "A framework for designing the architectures of deep convolutional neural networks," *Entropy*, vol. 19, no. 6, 2017.

[5] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*. ACM, 2012.

[6] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail," in *2012 IEEE Symposium on Security and Privacy*. IEEE, 2012.

[7] L. Finley, "VPNs won't save you from congress' Internet privacy giveaway," https://www.wired.com/2017/03/

vpns-wont-save-congress-internet-privacy-giveaway/, 2017, accessed: 2017-11-30.

[8] B. Fung, "What to expect now that Internet providers can collect and sell your Web browser history," https://www.washingtonpost.com/news/the-switch/wp/2017/03/29/what-to-expect-now-that-internet\-providers-can-collect-and-sell-your-web-browser-history/?utm_term=.5c52ff09c2be, 2017, accessed: 2017-11-30.

[9] J. Hayes and G. Danezis, "k-Fingerprinting: A robust scalable website fingerprinting technique," in *USENIX Security Symposium*, 2016, pp. 1187–1203.

[10] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*. ACM, 2009.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[12] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, "A critical evaluation of website fingerprinting attacks," in *Proceedings of the 2014 ACM Conference on Computer and Communications Security*. ACM, 2014.

[13] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *European Symposium on Research in Computer Security*. Springer, 2016, pp. 27–46.

[14] A. Ng, "CS229 lecture notes," http://cs229.stanford.edu/notes/cs229-notes3.pdf.

[15] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at Internet scale," in *The Network and Distributed System Security Symposium (NDSS)*, 2016.

[16] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, "Website fingerprinting in onion routing based anonymization networks," in *Proceedings of the 10th annual ACM Workshop on Privacy in the Electronic Society*. ACM, 2011, pp. 103–114.

[17] M. Perry, "Experimental defense for website traffic fingerprinting," *Tor project blog.*, 2011, https://blog.torproject.org/blog/experimental-defense-website-traffic-fingerprinting.

[18] ——, "A critique of website traffic fingerprinting attacks," *Tor project blog.*, 2013, https://blog.torproject.org.

[19] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen, "Automated website fingerprinting through deep learning," in *Proceedings of the 25th Network and Distributed System Security Symposium*. Internet Society, 2018.

[20] V. Shmatikov and M.-H. Wang, "Timing analysis in low-latency mix networks: Attacks and defenses," *European Symposium on Research in Computer Security*, pp. 18–33, 2006.

[21] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," *arXiv preprint arXiv:1801.02265*, 2018.

[22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[23] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, "Effective attacks and provable defenses for website fingerprinting," in *USENIX Security Symposium*, 2014, pp. 143–157.

[24] T. Wang and I. Goldberg, "Improved website fingerprinting on Tor," in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*. ACM, 2013.

[25] ——, "Walkie-Talkie: An efficient defense against passive website fingerprinting attacks," in *USENIX Security Symposium*, 2017, pp. 1375–1390.

[26] R. Zemel, R. Urtasun, and S. Fidler, "CSC 411: Lecture 05: nearest neighbors," https://www.cs.toronto.edu/~urtasun/courses/CSC411_Fall16/05_nn.pdf.